Exhibit 22

From: Thomas Kurian <thomas.kurian@oracle.com>
Sent: Mon Mar 22 2010 21:45:12 PDT
To: larry.ellison@oracle.com <larry.ellison@oracle.com>;safra.catz@oracle.com <safra.catz@oracle.com>
CC: andrea.nordemann@oracle.com <andrea.nordemann@oracle.com>;joyce.higashi@oracle.com <joyce.higashi@oracle.com>;hasan.rizvi@oracle.com <hasan.rizvi@oracle.com>;heidi.bielanski@oracle.com <heidi.bielanski@oracle.com>
Subject: For your meeting with Eric Schmidt
Attachments: javaUpdate.ppt;javaAPI.xls

Importance: Normal
Priority: Normal
Sensitivity: None

Larry - I understand that you will be meeting with Eric Schmidt on Wednesday. To give you some background for the meeting we have prepared two documents for you. Here is a quick summary of the information.

Background on how things work:

A. Today when you write Java code:

(i) You can use a variety of Java editors (Eclipse, JDeveloper, NetBeans and others) to write Java Source Code; (ii) the source code in turn calls a set of programming APIs called class libraries (which provide access to various functions); (iii) to run, the Java source gets compiled to a runtime form called Java bytecode that gets executed by a Java Virtual Machine. (iv) There are 3 types of Java Virtual Machines - HotSpot which is the desktop and server VM; CDC which is the Java ME Virtual Machine for blu-ray and TV type devices; and CLDC which is the Java ME Virtual Machine for traditional phones.

B. Today when you write Android code:

(i) You can use one editor only - Eclipse; (ii) the source code in turn calls a set of class libraries - some/a significant number of these class libraries were taken from Apache Harmony; there are several others that were defined and built specifically focused on Mobile Phone functionality by Google itself; (iii) to run, the Android source gets converted to a compact bytecode representation called dex bytecode. (iv) Finally, the dex bytecode executes by a single Android VM called the Dalvik Virtual Machine.

Performance Comparison

Note: the Java VM that most companies would consider running on the class of devices that Android runs on is the CDC VM or Hotspot not the CLDC VM that is targeted to lower powered phones. Note based on tests that the Sun team has done in detail, the CDC VM is faster by 2X on average and can be as much as 3X faster from a runtime point of view depending on the application than the Android Dalvik VM. The start up performance issue which has sometimes been stated as a problem is also addressed with the JDK7 release. Note that Android does not really compile the code but runs interpreted - they have a very early Just in Time compiler but Java has a very mature JIT and Ahead of Time (AOT) compiler which are useful for different types of optimizations. These are the reasons for the superior

performance.

Processor and Language Portability

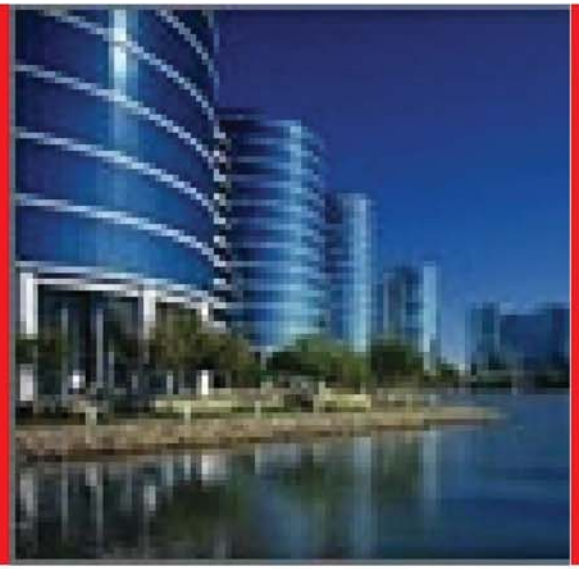
Today, Android only runs on the ARM Processor and X86. The JavaVM also runs on a large number of platforms (especially Processors) giving Android a much more portable programming model if they want to go to extend their OS play to other processors especially to devices such as Blu Ray, TVs etc. Android also only runs applications specifically built to the Android Yava language specification. It does not run any other languages - the Java VM can run applications written not just in Java but because of JSR 286 other languages especially Javascript and Groovy. The Javascript issue is important since it is important for browser Applications and Javascript is one of the key pieces that Google promotes for its Chrome browser/OS - ironically the Android Dalvik VM does not run Javascript. The reason that this is important to Google is that there are nearly 14 variants of Android out in the market and it is in their interest to keep one reference version as the primary version and they can only do this if they port and attract other languages to it.

Please note: (i) There are more details in the attached slides backing these points. (ii) The spreadsheet also captures the differences in class libraries between Android and Java.

Please let me know if you need anything further for the meeting.

Thank you

Thomas



ORACLE

Java on Android Devices

Background

- We have two JavaME platforms that are available on mobile devices
  - CLDC (Connected Limited Device Configuration)
    - Feature phones; biggest deployment (CLDC JVM = Monty)
  - CDC (Connected Device Configuration)
    - High end phones (CDC JVM = CVM)
- We are currently also working on putting JavaSE on mobile and embedded devices
  - High end phones (SE JVM = Hotspot)
- We will be using CDC for most of our comparisons
- CDC is Java 1.4 and Android is Java 1.5

ORACLE

Key Java advantages

- Runtime speed: 2-3x faster
- Support for multiple chipsets and OS'es
- Proven fine grain security model
  - Scales from mobile to desktop
- Future
  - JavaSE supports multiple languages with competitive performance (especially JavaScript). It also supports Ruby, Scala and Groovy. We are bringing this to the mobile.

ORACLE

VM (1/2)

- Runtime performance
  - JIT and AOT compilers can enable faster performance of pure Java code
  - Dalvik has a prototype JIT for ARM, but appears to be about 1/3 the speed of what Sun offers
  - Sun VMs do many more optimizations in JIT than the Dalvik bytecode to dex converter
    - Dalvik performs very little online translation, has a compact format (dex) which is directly interpreted at reasonable speed/power
    - Because it is an interpreted format, it can be compact but it is slower
    - CDC can also address the compact issue. We have back-burner experiments with code caching, fast-compile formats (zapcode), and compressed loading formats.
    - These could be combined with similar engineering tradeoffs to minimize startup translation and execution cycles

ORACLE

VM (2/2)

- Runtime performance
  - Dalvik has a faster native method interface. We have similar technology that we could use to improve performance.
- Multi-lingual support (Java SE) – Java and Javascript
  - We ran a pilot project on 12/2009 showing that our runtime can perform as well as V8 (Google JS VM), with some tuning.
  - A general-purpose VM running JavaScript and Java has more value to customers especially in combination with supporting a browser

ORACLE

Footprint

- Static Footprint
  - No Java advantage here. A Dalvik stack should be smaller than a Java stack due to the DEX bytecode format. That's the nature of DEX vs Java bytecodes. We have a solution that we can use to address this
- RAM Footprint: The following three technologies can affect RAM footprint:
  - GC (Garbage Collection) - dalvik's GC does not compact => heap fragmentation
  - MTASK (Multi-task) - CDC and Dalvik have similar process based MTASK functionality.
  - MVM (Multi-tasking VM)- CLDC and SE (prototype) have single process MVM functionality. Uses much less memory per application, but does not provide the same isolation and security as MTASK.
- CDC minimum footprint smaller than Dalvik

ORACLE

Portability

- CPU ports
  - CDC - ARM, MIPS, PowerPC, Sparc, x86, ST20, SH3/4, AM34, ARC
  - SE - ARM, x86, PowerPC, Sparc
  - Dalvik - ARM, x86
- OS
  - CDC - Linux, Solaris, Mac OS X, iPhone, NetBSD, OS20. Win32. WM/WinCE, Brew. Symbian, uCos, VxWorks, NetBSD
  - SE- Linux, XP Embedded, Linux GC, Windows, Solaris, NetBSD, QNX
  - Dalvik - Linux

ORACLE

APIs

See spreadsheet

- Java runtime libraries
  - Android uses a subset of Java runtime libraries they picked from Apache Harmony
- Android libraries
  - Android has built several additional packages that provides extensive support for device functionality
  - Java has equivalent libraries for most with the major exception of
    - Telephony, speech and database

ORACLE

Dev environment

- Java Tooling
  - Supported for multiple developer tools, Eclipse, Netbeans, JDeveloper, IntelliJ
  - JVMTI (JVM Tools Interface)
    - Support the standards for debugging JVMTI and for profiling JVMPI
    - Android doesn't have JVMTI, but does integrate with the Eclipse IDE
  - Extensive GUI for app profiling & heap examination - Visual VM
- Android/Dalvik Tooling
  - Tight integration with Eclipse IDE

ORACLE

Appendix

ORACLE

Performance

- Startup performance
  - CDC and Monty feature for system classes (Romization)
    - ~ Faster than Dalvik startup time (need to measure exact times)
  - CLDC Feature (Monet)
    - Runtime romizing for midlets.
    - Once a midlet is downloaded, it is romized at runtime so in the future it can just be mapped into memory.
    - Greatly speeds up the loading of midlets. ~ Faster than Dalvik.
  - AOT - CDC and CLDC feature
    - Speeds startup by removing the CPU time it takes to compile methods.
  - With JDK7 (not shipped yet) we have significantly improved startup speed and in combination with compressed loading formats, we are competitive in startup performance

ORACLE